Widyariset | Vol. 3 No. 1 (2017) Hlm. 19 - 34

Implementasi Mikrokontroler *Single Master - Multi Slave* untuk Pengiriman dan Penerimaan Data pada Prototipe Stasiun Cuaca

Single Master - Multiple Slaves Microcontroller Implementation for Sending and Receiving Data on Weather Station Prototype

Dikdik Krisnandi[1,*] dan Purnomo Husnul Khotimah[1]
[1]Pusat Penelitian Informatika, Lembaga Ilmu Pengetahuan Indonesia, Jl. Cisitu 21/154D Komplek LIPI, Bandung, Indonesia
[*]*E-mail*: dikdik@informatika.lipi.go.id

ARTICLE INFO

Article history
Received date:
14 June 2016
Received in revised form date:
10 September 2016
Accepted date:
19 October 2016
Available online date:
31 May 2017

Abstract

We have carried out the microcontroller research implementation as a Master to control multiple Slaves microcontroller on weather station prototype. Master Microcontroller could be interconnected with four Slaves microcontrollers Slaves, in which each has a different type of input. In this research, we used an ATmega8535 microcontroller. To distinguish between a Slave microcontroller and another Slave microcontroller, an addressing system (ID) is used. Data communication used a standard serial port RS-232 and RS-485 ports, the connector is used to connect the two ports. The Single Board Controller (SBC) Master serves user commands from users, sending the commands and processing Slaves. User commands are given via the keypad and then transmitted using a Universal Asynchronous Receiver-Transmitter (USART) to SBC Slaves. The process on the SBC Slaves is waiting for orders from the Master. The order is detected using an interrupt. After that, the commands are processed and the respond is sent to the SBC Master via USART. When there is no command, Slaves do the idle process. The result shows that the microcontroller application single Master - multiple Slaves has functioned for sending and receiving data in accordance with the specified command. This is shown by the output on a display that has shown the result as expected. The power consumption of each SBC is relatively small which is 0,745 Watt. Thus, it makes the system more economically profitable.

Keywords: *Microcontroller, ATmega8535, Master, Slave, Single board controller*

DOI: http://dx.doi.org/10.14203/widyariset.3.1.2017.19-34

Kata kunci:

Mikrokontroler
ATmega8535
Master
Slave
Single board controller

Abstrak

Kami telah melakukan penelitian yang berkaitan dengan implementasi mikrokontroler sebagai *Master* untuk pengendali mikrokontroler multi *Slave* pada prototipe stasiun pemantau cuaca. Mikrokontroler *Master* dapat saling berhubungan dengan empat mikrokontroler *Slave* dimana keempat *Slave* tersebut mempunyai jenis *input* yang berbeda. Pada penelitian ini, mikrokontroler yang digunakan adalah jenis ATmega8535. Untuk membedakan antara satu mikrokontroler *Slave* dengan mikrokontroler *Slave* yang lain, digunakan sistem pengalamatan (*ID*). Komunikasi data menggunakan standar serial *port* RS-232 dan *port* RS-485, dimana untuk menghubungkan kedua *port* tersebut digunakan konektor. *Single Board Controller* (*SBC*) Master berfungsi untuk menerima perintah dari pengguna, mengirimkan perintah pengguna dan memroses respon dari *SBC Slave*. Perintah pengguna diberikan melalui keypad yang kemudian dikirimkan menggunakan *Universal Synchronous Receiver-Transmitter* (*USART*) kepada *SBC Slave*. Proses pada *SBC Slave* adalah menunggu perintah dari *SBC Master*. Perintah tersebut dideteksi dengan menggunakan interupsi. Setelah itu perintah diproses dan respon dikirimkan kepada *SBC Master* melalui *USART*. Ketika tidak ada perintah, *Slave* melakukan proses *idle*. Hasil pengujian menunjukkan bahwa fungsi aplikasi mikrokontroler single *Master* - multi *Slave* untuk pengiriman dan penerimaan data telah berjalan dengan baik sesuai dengan spesifikasi peri tah. Hal tersebut ditunjukkan dengan output pada *display* yang telah menu njukkan hasil sesuai dengan yang diharapkan. Penggunaan daya setiap *SBC* relatif kecil, yaitu 0,745 Watt sehingga sistem lebih menguntungkan secara ekonomis.

PENDAHULUAN

Mikrokontroler banyak digunakan dalam perkembangannya saat ini, tidak terkecuali pada stasiun pemantau cuaca. Hanya saja mikrokontroler biasanya bekerja secara mandiri atau tidak berhubungan antara satu mikrokontroler dengan mikrokontroler yang lain. Tujuan penelitian ini adalah mengembangkan sistem komunikasi *Master-Slave* untuk mempermudah pengguna mengirim dan mengambil data dari dan ke banyak mikrokontroler *Slave* dengan menggunakan satu mikrokontroler *Master* yang diterapkan pada prototipe stasiun pemantau cuaca. Pada dasarnya, mikrokontroler dapat menerima *input* berupa sinyal digital dan analog yang berupa tegangan. Pada penelitian ini mikrokontroler akan menerima analog *input*, digital *input*, pulsa digital *input*, dan sebagai digital *output*. Dengan harapan aplikasi sistem yang dibuat akan sangat fleksibel untuk pengembangan sistem yang lebih besar dan kompleks. Hal ini dikarenakan sistem yang dibuat memungkinkan berbagai macam *input* dan *output*.

Manfaat dari penelitian ini adalah diperolehnya prototipe sistem komunikasi *single Master* dan multi *Slave micro controller* untuk pengiriman dan penerimaan data ke dan dari *multipoint* melalui s*ingle bus* yang berupa kabel. Beberapa sensor dan aktuator yang banyak digunakan pada stasiun cuaca digunakan sebagai *input*. Hal tersebut diharapkan dapat meningkatkan kinerja sistem secara keseluruhan dan memberikan nilai tambah dari segi ekonomis pada proses otomatisasi.

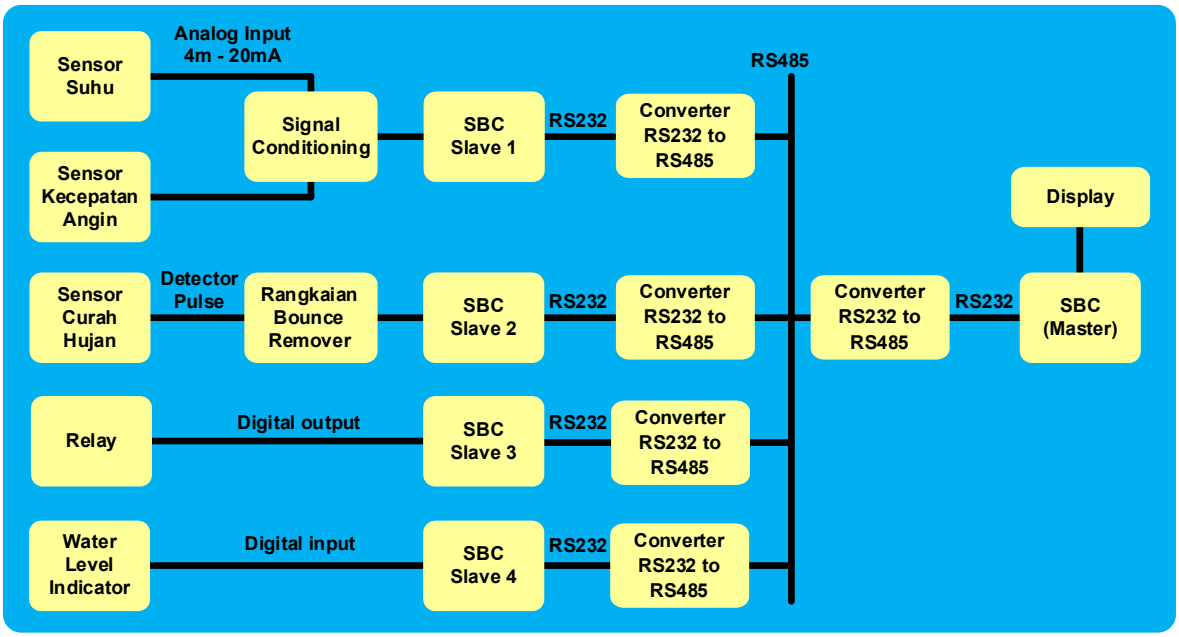
Ada beberapa penelitian tentang aplikasi mikrokontroler dengan arsitektur *Master - Slave* dengan fokus yang berbeda. Di antaranya adalah Hsiung (2007) yang membuat aplikasi mikrokontroler dengan menggunakan mikrokontroler PIC16F84 untuk *multiple motor direct current* (DC). Akan tetapi dibandingkan dengan mikrokontroler Atmel AVR, jenis mikrokontroler PIC lebih mahal harganya dan *programming interface*-nya lebih rumit jika dibuat sendiri.

Sankar dkk. (Sankar, Tiong, and Koh 2009) membuat mikrokontroler *Master* sebagai pengendali yang dapat ber komunikasi dengan mikrokontroler *Slave* melalui *bus* serial menggunakan modul *Universal Asynchronous Receiver-Transmitter* (USART). Sankar dkk. meng gunakan mikrokontroler PIC18F4550 pada penelitian tersebut.

Altaf and Iqbal (2010) membuat aplikasi satu mikrokontroler *Master* dan dua mikrokontroler *Slave* dengan menggunakan mikrokontroler Atmel 89C51. Pada penelitian ini digunakan mikrokontroler Atmel AVR. Jika dibandingkan dengan mikrokontroler 89C51, mikrokontroler AVR memiliki keunggulan yakni memiliki internal EPROM, memiliki delapan *channel* ADC 10 bit dan satu instruksi hanya dilaksanakan satu *clock*. Hal ini berbeda dengan mikrokontroler 89C51 yang membutuhkan 6-12 *clock*. Dibandingkan dengan mikrokontroler berbasis *Advanced RISC Machine* (ARM), kemampuan mikrokontroler AVR sudah memadai untuk mengeksekusi perintah pada sistem yang dibuat karena tidak membutuhkan lokasi *address* yang besar dan tidak memerlukan kecepatan *clock* yang sangat tinggi. Hal terpenting lainnya adalah pada *development tools*-nya, yakni *cross compiler* (C) yang dapat digunakan secara gratis.

BAHAN DAN METODE

Gambar 1 memperlihatkan desain sistem yang dibuat. Mikrokontroler tersebut dirangkai dalam bentuk *Single Board Controller* (SBC) yang merupakan modul



Gambar 1. Desain sistem yang dibuat

mikrokontroler, baik itu sebagai *Master* ataupun sebagai *Slave*.

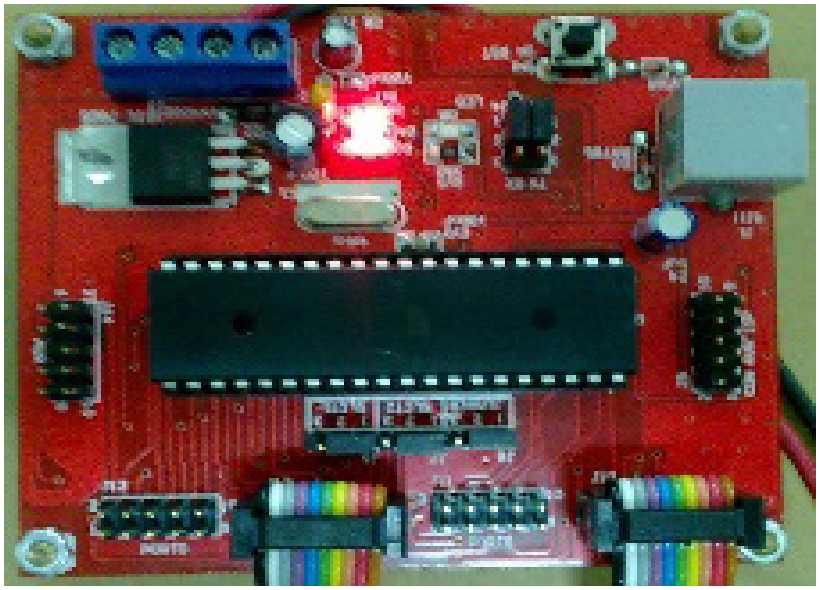
Fungsi dari SBC *Master* adalah sebagai koordinator setiap SBC *Slave*, *interface* akses data, memberikan *command* ke SBC *Slave*, menampilkan data ke *display,* dan menyediakan suatu format data untuk diteruskan ke *data logger* (Khotimah and Krisnandi 2010). Sedangkan fungsi SBC *Slave* adalah melayani modul *Input/Output* (I/O). Supaya SBC *Master* dapat membedakan antara satu SBC *Slave* yang satu dengan SBC *Slave* yang lain, digunakan sistem pengalamatan (ID) pada setiap SBC *Slave*. Sedangkan komunikasi datanya menggunakan standar *port* serial RS-232 dan *port* RS-485. Sebagai jembatan kedua standar *port* tersebut digunakan RS-232 *to* RS-485 *converter*.

Bagian Mikrokontroler *Master* dan *Slave*

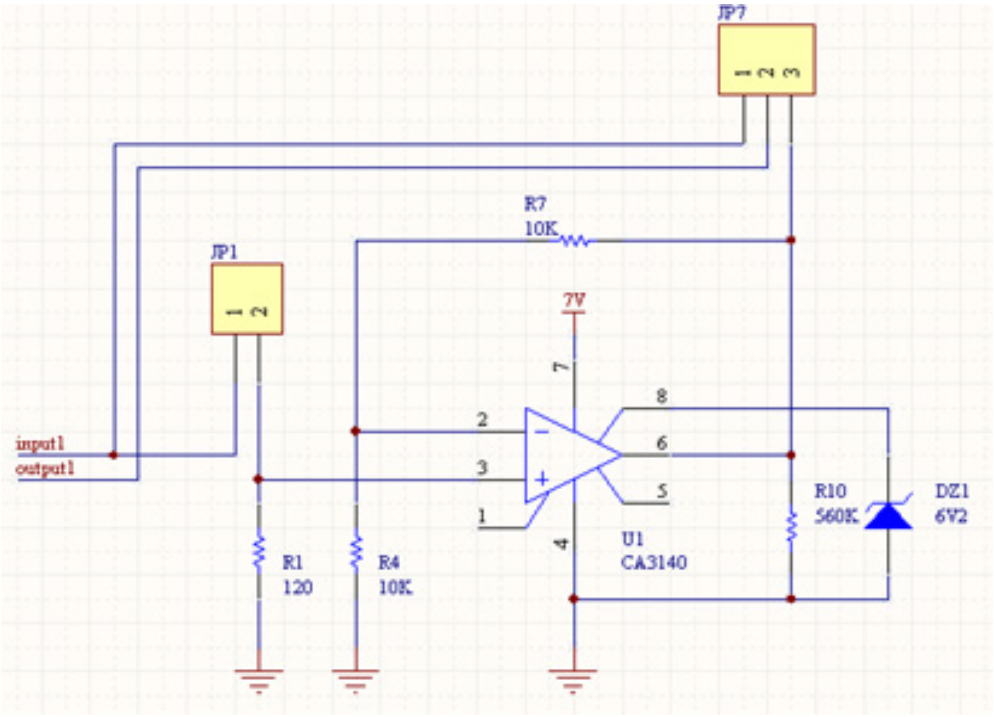
SBC *Master* merupakan pusat pengendali SBC *Slave-Slave* yang ada. SBC *Master* menggunakan modul DT-AVR *low cost micro system* dengan mikrokontroler Atmel AVR ATmega (Barrett 2010; Gadre 2001). SBC *Master* dan SBC *Slave* sama-sama menggunakan modul DT-AVR *low cost micro system* dengan mikrokontroler ATmega8535. Modul tersebut memiliki kemampuan untuk melakukan komunikasi data serial melalui USART RS-232. Frekuensi osilator yang digunakan 4 MHz dengan *baud rate* 9.600 bps (Atmel 2013). Gambar 2 memperlihatkan modul DT-AVR dengan mikrokontroler ATmega8535 yang digunakan sebagai SBC *Master* maupun SBC *Slave*.

SBC *Slave* 1

SBC *Slave* 1 menggunakan sinyal *analog* sebagai *input*nya. Sensor suhu (WE700) dan sensor kecepatan angin (WE550) yang digunakan akan menghasilkan sinyal analog berupa arus 4 mA – 20 mA. Sementara itu modul mikrokontroler ATmega8535 hanya dapat menerima *input* berupa tegangan 0 V - 5 V. Untuk menyelesaikan masalah tersebut digunakan suatu rangkaian *signal conditioning* atau pengondisi sinyal. Komponen yang digunakan dalam *signal conditioning* adalah menggunakan resistor sebagai pengkonversi dari arus menjadi tegangan dan dihubungkan dengan sebuah *operational amplifier* (*op-amp*) CA314 yang berfungsi sebagai *non-inverting buffer* (Boylested and Nashelsky 2009).



Gambar 2. DT-AVR *Low Cost Micro System* dengan Mikrokontroler ATmega 8535



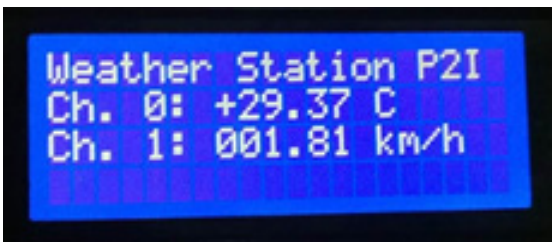
Gambar 3. Skematik rangkaian *signal conditioning* yang digunakan

Output dari sensor suhu dan sensor kecepatan angin yang berupa arus antara 4 mA - 20 mA akan dilewatkan pada resistor 120 Ω sehingga hasilnya menjadi tegangan *input* (V_i). Nilai (V_i) ini akan masuk ke *non-inverting buffer* yang mempunyai penguatan 2x. Tegangan hasil penguatan tersebut berkisar antara 0,9 V - 4,8 V sehingga sudah memenuhi syarat untuk menjadi *input* mikrokontroler. *Signal conditioning* yang dibuat terdiri enam buah kanal (Krisnandi 2011). Gambar 3 menunjukkan skematik rangkaian *signal conditioning* yang di gunakan.

a. Sensor Suhu (WE700) b. Sensor Kecepatan Angin (WE550)

Gambar 4. Sensor Suhu dan Sensor Kecepatan Angin pada SBC *Slave* 1

Dari hasil percobaan, SBC *Slave* 1 yang digunakan ternyata dapat berfungsi dengan baik. Ini dapat dilihat dari tampilan pada *Liquid Crystal Display* (LCD) yang menunjukkan perubahan nilai sesuai dengan yang diharapkan. Sebagai pembanding, untuk hasil SBC *Slave* 1 dari *output* sensor suhu, kami menggunakan 2 buah termometer ruangan, yakni yang bertipe analog maupun *digital*. Terlihat bahwa tidak ada

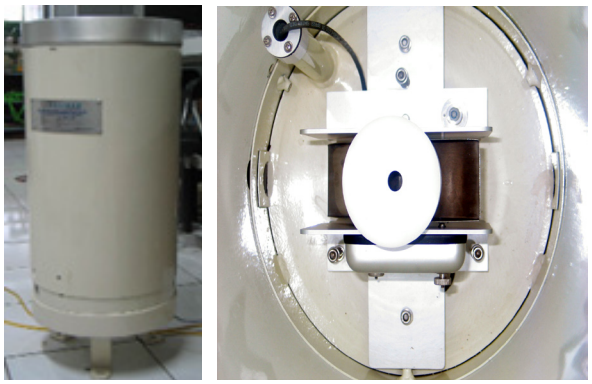


Gambar 5. Tampilan hasil pada LCD di SBC *Slave* 1

perbedaan nilai yang signifikan dari ketiga alat tersebut. Untuk sensor kecepatan angin, kami menggunakan kipas angin untuk melakukan simulasinya.

SBC *Slave* 2

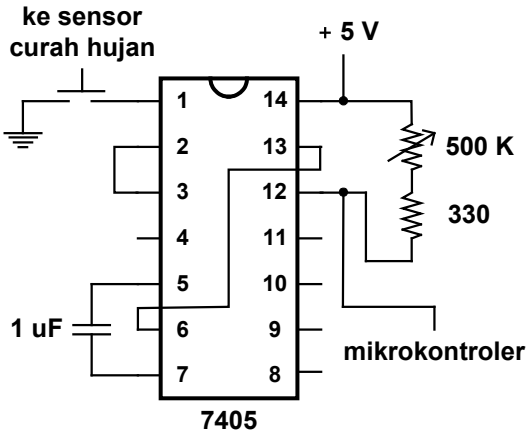
Untuk SBC *Slave* 2, *input* menggunakan sensor curah hujan dengan tipe *tipping bucket* model LK-5050-RG yang menghasilkan sinyal jenis *pulse*. SBC Master diharapkan mampu mendeteksi jumlah *pulse* yang dikeluarkan oleh sensor. Sensor tersebut menggunakan prinsip penampung berjungkit. Penampung berjungkit terdiri dari dua buah jungkitan yang dapat bergulir dan terletak di bawah corong dalam pengumpul dan dirancang untuk mengukur



Gambar 6. Sensor curah hujan tipe *tipping bucket* LK-5050-RG

kenaikan curah hujan 1 mm. Pada saat volume air sekitar 70,7 ml maka penampung berjungkit mengisi titik dimana ia akan menjungkit lagi.

Untuk menghindari kesalahan jumlah pembacaan *pulse* yang masuk (*bouncing*) pada mikrokontroler digunakan rangkaian *bounce remover*. Penyebab terjadinya bouncing adalah karena terjadinya getaran pada kontak saklar di *tipping bucket rain gauge* akibat adanya momentum elastisitas beraksi yang bersamaan. Alasan penangan an secara *hardware* (menggunakan rangkaian *bounce remover*) adalah karena jika hanya ditangani secara *software* ternyata

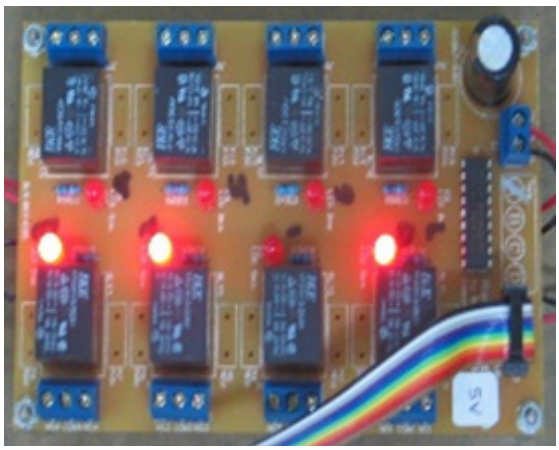


Gambar 7. Rangkaian *bounce remover* yang digunakan

tidak mampu mendeteksi perubahan logika yang lebih cepat dari *delay* program.

SBC *Slave* 3

SBC *Slave* 3 merupakan modul *relay* dengan delapan buah *relay* HRS4H-S-DC5V sebagai targetnya. Pada penelitian ini hanya digunakan empat buah *relay* saja. Keempat *relay* tersebut masing-masing tersambung dengan *Light Emitting Diode* (LED) yang akan menjadi indikator. Jika LED menyala maka *relay* tersebut aktif sedangkan jika LED tidak menyala maka dapat dipastikan bahwa *relay* tersebut sedang tidak bekerja. SBC *Master* diharap kan mampu mengendalikan *relay-relay*



Gambar 8. Modul *Relay* yang digunakan

tersebut. Modul *relay* digunakan sebagai bukti bahwa SBC *Master* dapat mengendali kan digital *output*.

SBC *Slave* 4

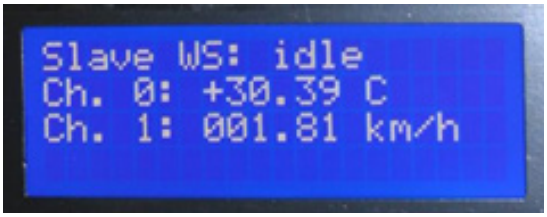
SBC *Slave* 4 merupakan indikator level air yang bekerja seperti sebuah *switch*. Indikator level air digunakan untuk membuktikan bahwa SBC *Master* dapat juga digunakan untuk digital *input*.

RS-232 *to* RS-485 *Converter*, *Display*, dan *Keypad*

RS-232 *to* RS-485 *converter* merupakan suatu pengubah level tegangan dua arah antara RS-232 dan RS-485. Rangkaian ini dapat berfungsi sebagai jalur komunikasi antar mikrokontroler yang berantar-muka (ber-*interface*) USART RS-232 atau ber antarmuka USART RS-485.

RS-485 merupakan antarmuka serial yang dikeluarkan oleh *Electronics Industries Association* (EIA) dan dirancang untuk komunikasi data kecepatan tinggi dan memungkinkan konfigurasi *multipoint* (*party line*) (Goldie 1998). Saluran komunikasi *multipoint* ini dapat dihubungkan sampai dengan 32 *drivers/generators* dan 32 *receivers* pada *single* (*two wires*) *bus* serta pengenalan terhadap *repeater* dan *high-impedance drivers/receivers* secara otomatis. Keterbatasan jumlah *driver* dan *receiver* tersebut dapat diperluas hingga ratusan (bahkan ribuan) titik pada jaringan. RS-485 dengan menggunakan sistem transmisi sa luran ganda (*differential data transmission*) untuk transmisi saluran datanya. Dalam pemasangannya, digunakan dua kawat (*two wires*) untuk tiap sinyal. Level logika direferensikan oleh perbedaan potensial di antara dua kawat tersebut, bukan terhadap arde. RS-485 mempunyai sifat yang sangat kebal terhadap gangguan listrik dan dapat menyalurkan data hingga maksimum 4.000 kaki (= 1.219 m) dengan kecepatan 100 Kbps atau 40 kaki (= 12,192 m) dengan kecepatan 10 Mbps (maksimum data *rate*).

Display yang digunakan adalah LCD JHD 204A yang menampilkan 20 kolom dan empat baris untuk memperlihatkan tampilan telah sesuai dengan yang di inginkan atau belum. Sedangkan *keypad* diperlukan untuk berinteraksi dengan sistem. *Keypad* yang digunakan dalam penelitian ini berupa *push-button* dengan matrik 3 × 4.

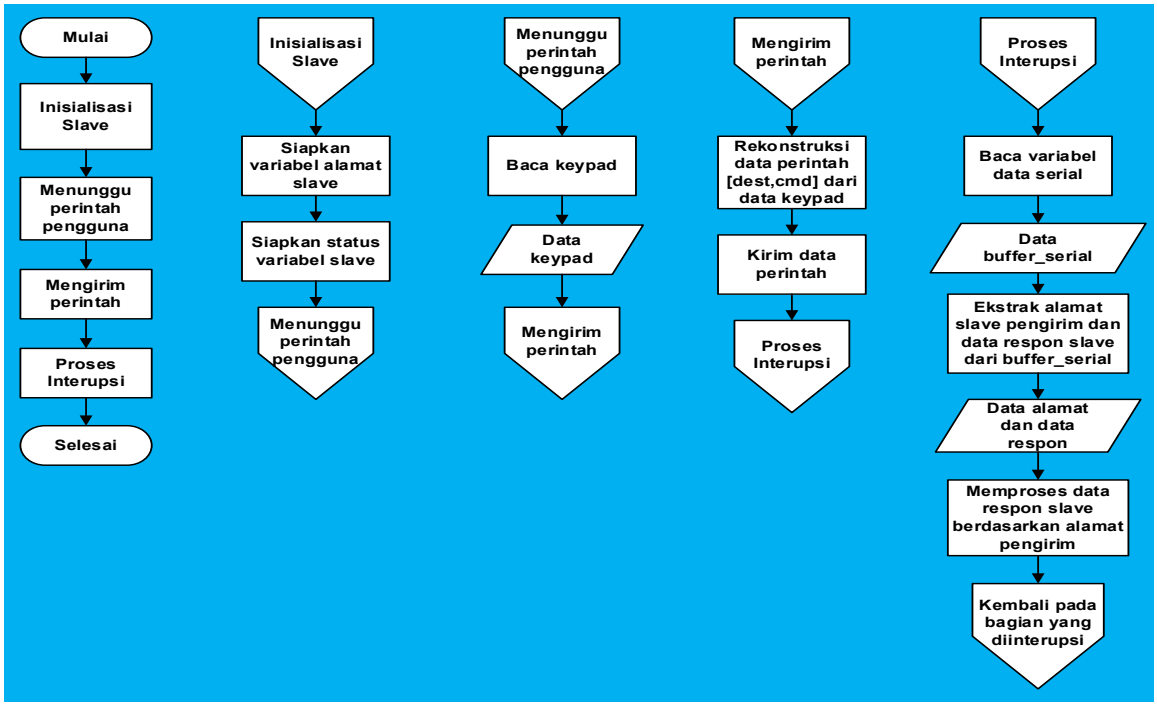


Gambar 9. LCD JHD 204A yang digunakan

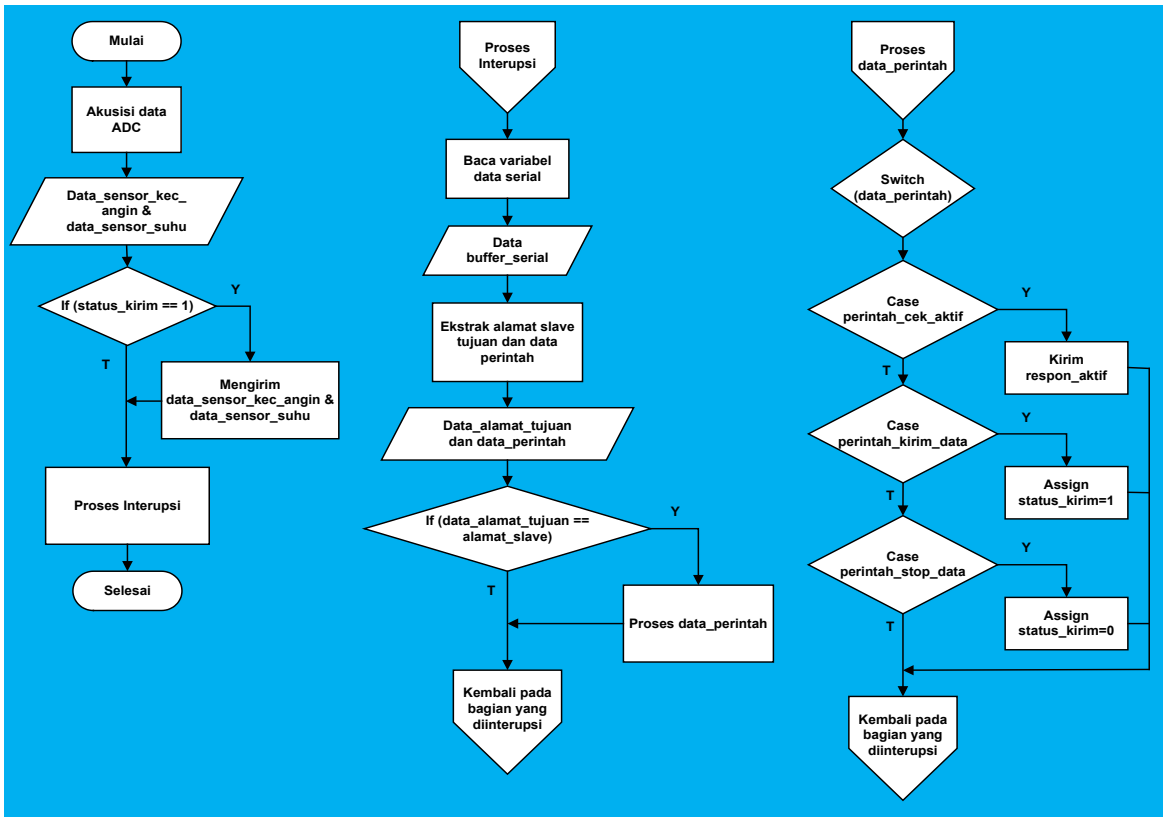
Komunikasi antara *Master* dan *Slave*

Komunikasi antar-SBC *Master* dan SBC *Slave* dilakukan seperti pada *flowchart* pada Gambar 10, 11, 12, 13 dan 14. *Software* menggunakan *Code Vision* AVR sebagai *compiler*-nya Pada Gambar 10, diperlihatkan *flowchart* SBC *Master*.

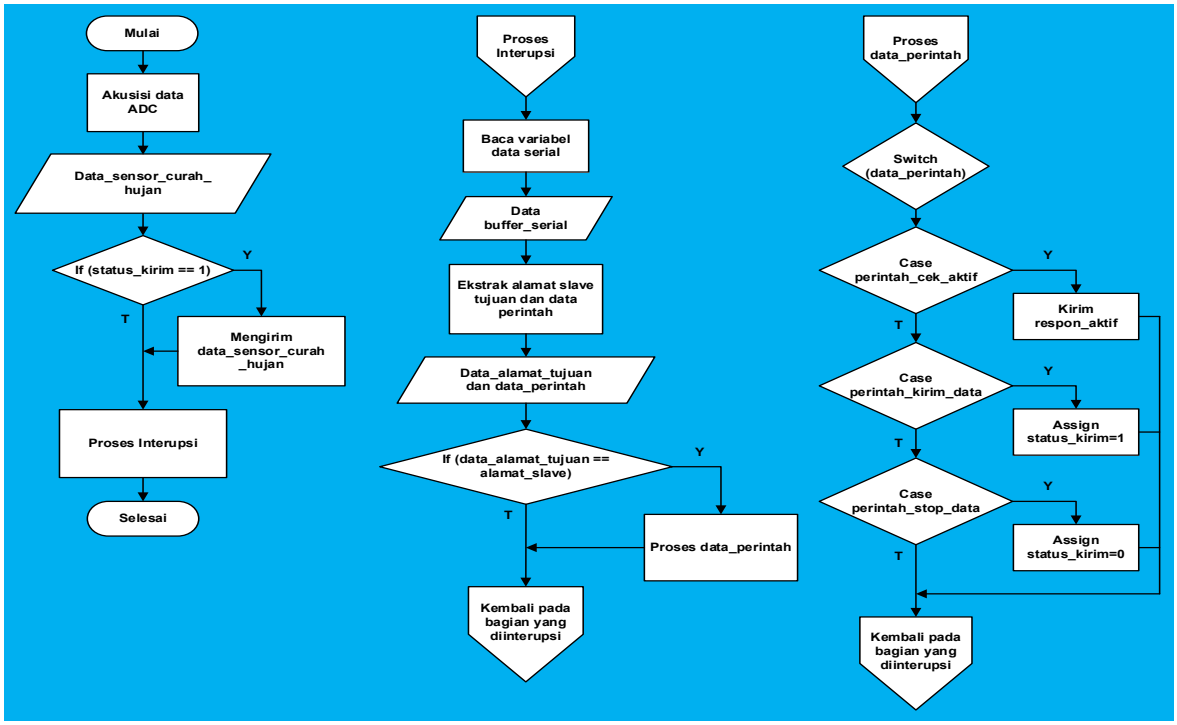
Seperti terlihat pada Gambar 10, pada *Master* terjadi empat proses utama, yaitu inisialisasi, menunggu perintah pengguna, pengiriman perintah, dan proses interupsi. Pada proses inisialisasi *Slave*, *Master* melakukan inisialisasi alamat dan status *Slave*. Pada aplikasi versi pertama ini, alamat diberikan melalui sebuah variabel alamat. Proses berikutnya adalah proses *idle* pada *Master*, yaitu menunggu perintah pengguna. Pada proses ini dilakukan pembacaan pada *keypad*. Data hasil pembacaan *keypad* diteruskan pada proses ketiga, yaitu pengiriman perintah. Pengiriman pe rintah terdiri atas rekonstruksi perintah dan pengiriman data perintah. Format perintah yang digunakan adalah [alamat_Slave_ tujuan, perintah].



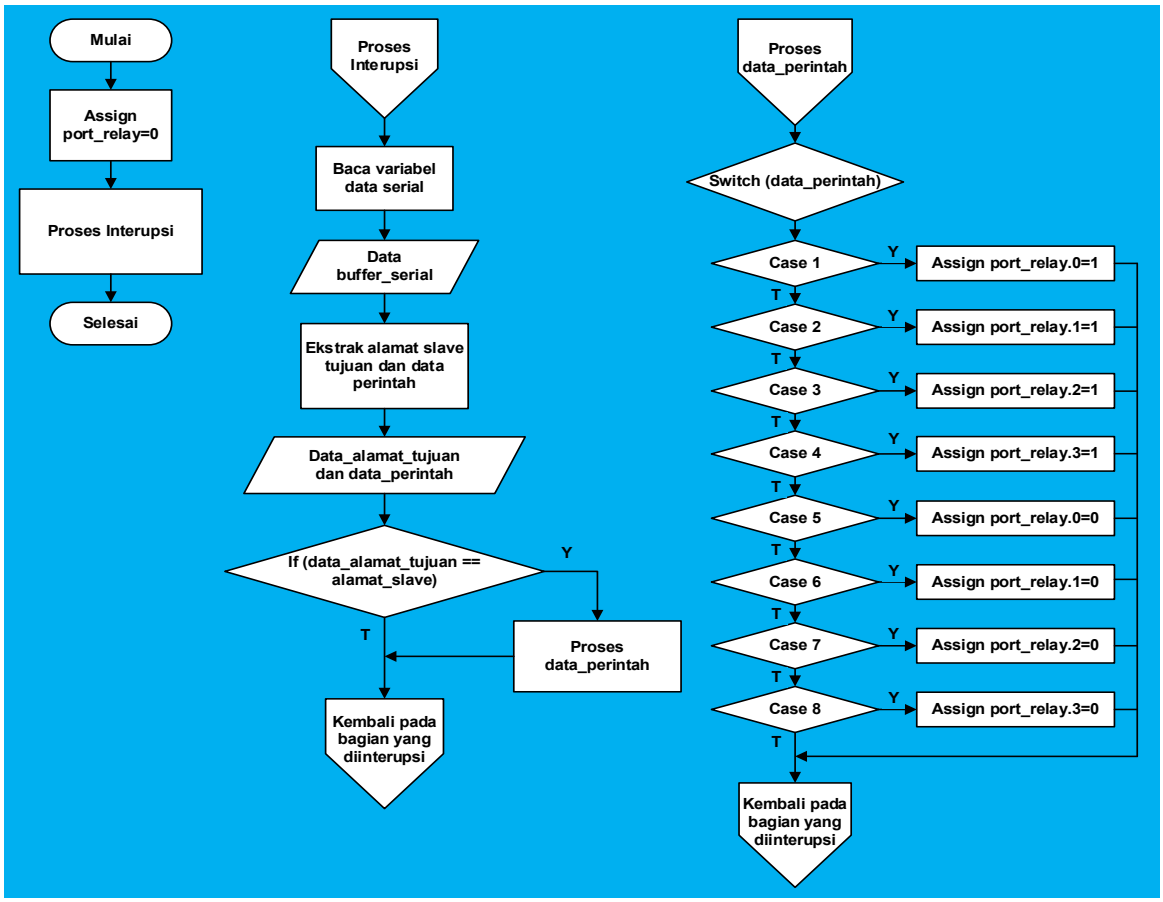
Gambar 10. *Flowchart* SBC *Master*



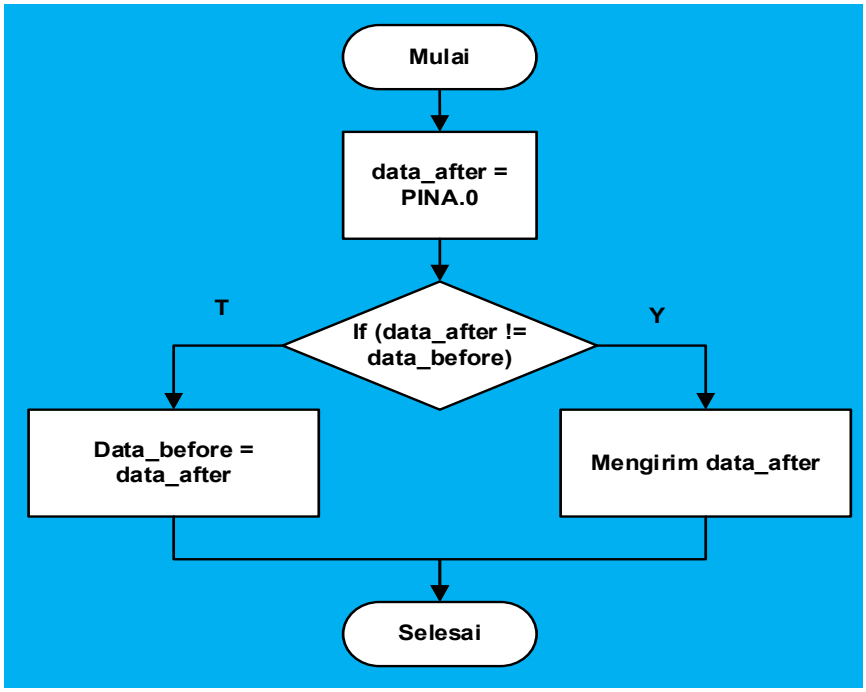
Gambar 11. *Flowchart* SBC *Slave* 1



Gambar 12. *Flowchart* SBC *Slave* 2



Gambar 13. *Flowchart* SBC *Slave* 3



Gambar 14. *Flowchart* SBC *Slave* 4

Pada pengambilan perintah dari pengguna digunakan teknik *pooling*, yaitu secara terus menerus mengecek masukan dari *keypad*. Sedangkan pada penerimaan data dari *Slave* digunakan interupsi, yaitu penghentian aliran program akibat ter jadinya pemicu tertentu dan memaksa eksekusi rutin / fungsi layanan interupsi, setelah selesai aliran program akan kembali ke pernyataan program sebelum terjadinya interupsi. Seperti terlihat pada *flowchart* SBC *Master* (Gambar 10), proses in terupsi diakhiri dengan kembali pada bagian yang diinterupsi. Proses penerimaan data dilakukan dengan membaca variabel data serial per *byte* dan memasukkan data tersebut pada *buffer*. Kemudian data *buffer* diekstrak untuk memperoleh alamat *Slave* dan data respon *Slave*. Proses berikutnya adalah memproses data respon sesuai dengan alamat asal (*Slave*).

Gambar 11, 12, 13, dan 14 mem-perlihatkan *flowchart* SBC *Slave* secara berurutan SBC S-50, S-51, S-52, dan S-53. SBC S-50 dan SBC S-51 memiliki fungsi yang sama, yaitu akusisi data sensor yang terhubung pada *Slave* dan merespon pe rintah dari *Master*. Oleh karena itu *flow-chart* pada Gambar 11 dan Gambar 12 memiliki kesamaan, yaitu mempunyai proses utama akusisi data, proses interupsi, dan proses data. Walaupun SBC S-52 memiliki fungsi yang agak berbeda yaitu mengontrol *relay* yang terhubung pada *Slave*, SBC S-52 memiliki *flowchart* yang sama dengan SBC S-50 dan SBC S-51 (terlihat pada Gambar 13).

Setiap perintah dari SBC *Master* akan direspon oleh SBC *Slave* yang diinginkan oleh *Master*.

Proses akusisi data adalah proses *idle Slave*, yaitu melakukan pembacaan pada *output* sensor. Proses interupsi pada *Slave* sama dengan *Master*, yaitu menangani penerimaan data yang dalam hal ini *Slave* menerima data dari *Master*. Proses pe nerimaan data terdiri dari pembacaan data serial dan menyimpannya ke *buffer*. Setelah itu data di-*buffer* diekstrak untuk memperoleh data alamat tujuan dan data perintah. Jika alamat tujuan sesuai den-gan alamat *Slave*, maka perintah akan di

terima oleh *Slave* dan diproses lebih lanjut. Kemudian proses data perintah dilakukan sesuai dengan perintahnya.

SBC S-53 memiliki *flowchart* yang cukup sederhana, yaitu hanya memiliki proses *idle* dan pengiriman data seperti terlihat pada Gambar 14. Tidak ada proses penerimaan data perintah dari *Master* karena SBC-S-53 hanya berfungsi sebagai modul *input* saja. Proses *idle* SBC S-53 adalah melakukan pembacaan pada pin A.0, dan ketika pin A.0 berubah nilai, maka *Slave* akan mengirimkan data ke *Master*.

HASIL DAN PEMBAHASAN

SBC *Master* yang berfungsi sebagai koordinator semua SBC *Slave* sekaligus memberikan perintah. Tabel 1 mem perlihatkan matrik perintah SBC *Master* pada *Slave.*

Tabel 1. Tabel perintah SBC *Master*

Key	SBC S-50 (alamat = 1)	SBC S-51 (alamat = 2)	SBC S-52 (alamat = 3)	SBC S-53 (alamat=4)
1	Cek aktif	Cek aktif	Nyalakan *relay* 1	-
2	Kirim data	Kirim data	Nyalakan *relay* 2	-
3	*Stop* data	*Stop* data	Nyalakan *relay* 3	-
4	-	-	Nyalakan *relay* 4	-
5	-	-	Matikan *relay* 1	-
6	-	-	Matikan *relay* 2	-
7	-	-	Matikan *relay* 3	-
8	-	-	Matikan *relay* 4	-

Dari Tabel 1 terlihat bahwa SBC *Slave* 4 (SBC S-53) tidak mempunyai perintah dari SBC *Master* karena SBS *Slave* 4 didesain hanya untuk secara otomatis me ngirimkan data apabila terjadi perubahan kondisi.

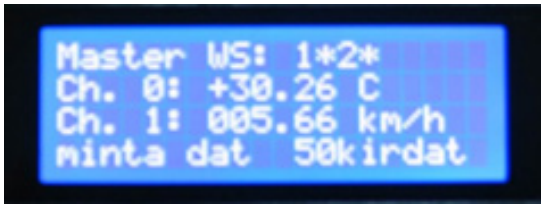
Berikut ini adalah langkah-langkah pengujian sistem komunikasi dan antara SBC *Master* dengan SBC *Slave* dan hasilnya. Pemasukan perintah interupsi menggunakan *keypad* dan ditampilkan pada *display*.

SBC *Master* dengan SBC *Slave* 1 (SBC S-50)

Langkah pengujian dan hasilnya pada *display*:

1. Pengguna memasukkan perintah 1*1* (cek aktif).
2. SBC *Master* mengirimkan perintah dan SBC S-50 mengirimkan respon.
3. Pengguna memasukkan perintah 1*2* (meminta mengirimkan data), SBC *Master* mengirimkan perintah dan SBC S-50 akan mengirimkan data (suhu dan kecepatan angin).
4. Pengguna memasukkan perintah 1*3* (meminta menghentikan pengiriman data), SBC *Master* mengirimkan pe rintah dan SBC S-50 akan menghenti-kan pengiriman data.

SBC *Master* dengan SBC *Slave* 2 (SBC S-51)



Gambar 15. *Display* saat SBC *Master* mengirimkan perintah dan SBC *Slave* 1 meresponnya dengan mengirimkan data

Langkah pengujian dan hasilnya pada *display*:

1. Pengguna memasukkan perintah 2*1* (cek aktif). SBC *Master* mengirimkan perintah dan SBC S-51 mengirimkan respon.
2. Pengguna memasukkan perintah 2*2* (meminta mengirimkan data), SBC

Master mengirimkan perintah dan SBC S-52 akan mengirimkan data (curah hujan).

3. Pengguna memasukkan perintah 2*3* (meminta menghentikan pengiriman data), SBC *Master* mengirimkan perintah dan SBC S-51 akan menghentikan pengiriman data.

Gambar 16. *Display* saat SBC *Master* mengirimkan perintah dan SBC *Slave* 2 meresponnya dengan menghentikan pengiriman data

SBC *Master* dengan SBC *Slave* 3 (SBC S-52)

Langkah pengujian hasilnya pada *display*:

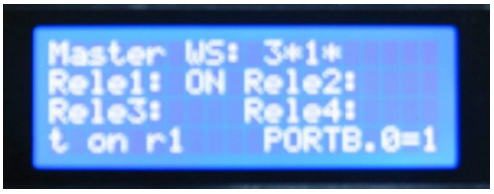
1. Pengguna memasukkan perintah 3*1* (menyalakan *relay* 1), maka SBC *Master* mengirimkan perintah dan SBC S-52 akan menyalakan *relay* 1.

2. SBC S-52 selain menyalakan *relay* 1 juga akan mengirimkan pesan kepada SBC *Master* untuk memberitahukan bahwa *relay* 1 telah berhasil dinyalakan (PORTB.0=1).

3. Jika pengguna memasukkan perintah 3*5* (mematikan *relay* 1), maka SBC *Master* mengirimkan perintah dan SBC S-52 akan mematikan *relay* 1.

4. SBC S-52 selain mematikan *relay* 1 juga akan mengirimkan pesan kepada SBC *Master* untuk memberitahukan bahwa *relay* 1 telah berhasil dimatikan (PORTB.0=0).

5. Perintah yang lain mengikuti tabel perintah *Master* seperti pada Tabel 1.



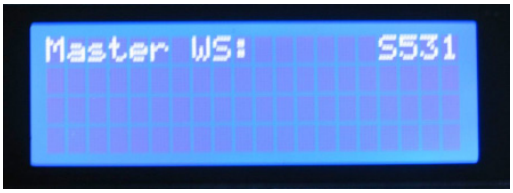
Gambar 17. *Display* saat SBC *Master* mengirimkan perintah dan SBC *Slave* 3 meresponnya dengan menyalakan *Relay* 1

SBC *Master* dengan SBC *Slave* 4 (SBC S-53)

Langkah pengujian hasilnya pada *display*:

1. Pengguna, dalam hal ini berupa indikator level air, jika air dalam keadaan kosong maka SBC S-53 akan mengirimkan karakter 0 dan SBC *Master* akan menampilkan hasilnya pada *display*.

2. Sebaliknya, jika air dalam keadaan penuh maka SBC S-53 akan mengirimkan karakter 1 dan SBC *Master* akan menampilkan hasilnya pada *display*.



Gambar 18. *Display* saat SBC *Slave* 4 merespon kondisi *Switch ON* dengan menampilkan angka 1

Oleh karena itu, data keseluruhan dari hasil percobaan dirangkum dalam Tabel 2.

Semua perintah pengguna diberikan melalui *keypad* yang kemudian dikirimkan menggunakan *Universal Asynchronous Receiver-Transmitter* (USART) kepada SBC *Slave*. Demikian pula sebaliknya dari SBC *Slave* menuju SBC *Master* juga menggunakan USART.

Panjang kabel yang digunakan pada penelitian ini sekitar 147 m. Dengan panjang kabel yang digunakan, ternyata hampir tidak ada *delay* yang terjadi. Dari

Tabel 2. Hasil Pengujian

No	Keypad	Perintah	Respon	Keterangan
SBC *Slave* 1 (SBC S-50)				
1	1*1*	cek aktif	50res akt	SBC *Slave* beralamat 50 merespon permintaan cek aktif *device*
2	1*2*	kirim data	50kirdat	SBC *Slave* beralamat 50 mengirim data ke SBC *Master*
3	1*3*	*stop* data	50*stop*	SBC *Slave* beralamat 50 menghentikan pengiriman data
SBC *Slave* 2 (SBC S-51)				
1	2*1*	cek aktif	51res akt	SBC *Slave* beralamat 51 merespon permintaan cek aktif *device*
2	2*2*	kirim data	51kirdat	SBC *Slave* beralamat 51 mengirim data ke SBC *Master*
3	2*3*	*stop* data	51*stop*	SBC *Slave* beralamat 51 menghentikan pengiriman data
SBC *Slave* 3 (SBC S-52)				
1	3*1*	nyalakan *relay* 1	Rele1: ON t on r1 PORTB.0=1	SBC *Slave* beralamat 52 merespon dengan menyalakan *relay* 1
2	3*2*	nyalakan *relay* 2	Rele2: ON t on r2 PORTB.1=1	SBC *Slave* beralamat 52 merespon dengan menyalakan *relay* 2
3	3*3*	nyalakan *relay* 3	Rele3: ON t on r3 PORTB.2=1	SBC *Slave* beralamat 52 merespon dengan menyalakan *relay* 3
4	3*4*	nyalakan *relay* 4	Rele4: ON t on r4 PORTB.3=1	SBC *Slave* beralamat 52 merespon dengan menyalakan *relay* 4
5	3*5*	matikan *relay* 1	Rele1:OFF t off r1 PORTB.0=0	SBC *Slave* beralamat 52 merespon dengan mematikan *relay* 1
6	3*6*	matikan *relay* 2	Rele2:OFF t off r2 PORTB.1=0	SBC *Slave* beralamat 52 merespon dengan mematikan *relay* 2
7	3*7*	matikan *relay* 3	Rele3:OFF t off r3 PORTB.2=0	SBC *Slave* beralamat 52 merespon dengan mematikan *relay* 3
8	3*8*	matikan *relay* 4	Rele1:OFF t off r4 PORTB.3=0	SBC *Slave* beralamat 52 merespon dengan mematikan *relay* 4
SBC *Slave* 4 (SBC S-53)				
No	Input		Output	Keterangan
1	*Switch OFF*		S530	SBC *Master* menampilkan di *display* kondisi *switch* yang belum terhubung
2	*Switch ON*		S531	SBC *Master* menampilkan di *display* kondisi *switch* yang telah terhubung

hasil pengukuran, daya yang digunakan untuk satu buah SBC adalah 0,205 Watt. Sedangkan daya untuk satu buah RS-232 *to* RS-485 *converter* sebesar 0,54 Watt. Dari hasil tersebut terlihat bahwa penggunaan daya dari satu buah aplikasi modul mikrokontroler ditambah daya *converter* yang digunakan relatif kecil, yaitu hanya 0,745 Watt. Hal ini menjadi nilai tambah sistem karena dengan jarak yang relatif jauh tapi konsumsi daya listriknya rendah sehingga sangat hemat energi.

KESIMPULAN

Pada penelitian ini telah dihasilkan sebuah prototipe sistem komunikasi *single Master* multi *Slave*. Berdasarkan hasil pengujian, komunikasi antara *Master* dan multi *Slave* dapat berjalan dengan baik. Hal tersebut membuktikan bahwa pemanfaatan mikrokontroler sebagai *Master* untuk mengendalikan mikrokontroler multi *Slave* mampu meningkatkan aplikasi penggunaan mikrokontroler yang selama ini bersifat mandiri menjadi dapat saling berkomunikasi menggunakan mikrokontroler *single Master* dan empat buah mikrokontroler *Slave* untuk pengiriman dan penerimaan data sehingga cocok untuk digunakan pada stasiun pemantau cuaca.

Penggunaan daya listrik pada satu buah aplikasi modul relatif kecil, yakni sebesar 0,745 Watt. Hal ini menunjukkan keuntungan ekonomis yang diperoleh dalam penggunaan sistem yang dibuat.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada seluruh tim *wireless weather station* di Pusat Penelitian Informatika LIPI atas kerjasama dan bantuannya dalam penelitian ini.

DAFTAR ACUAN

Altaf, Kashif, and Javaid Iqbal. 2010. "Multiprocessor Communication Using 8051 Microcontroller and RS-485 Line Driver."

Atmel. 2013. "8-Bit AVR with 8K Bytes In-System Programmable Flash ATmega8 ATmega8L." http://www.atmel.com/images/atmel-2486-8-bit-avr-microcontroller-atmega8_l_datasheet.pdf.

Barrett, Steven F. 2010. *Embedded Systems Design with the Atmel AVR Microcontroller – Part I*. Edited by Mitchell A. Thornton. Morgan & Claypool. doi:10.2200/S00138ED-1V01Y200910DCS024.

Boylested, Robert L., and Louis Nashelsky. 2009. *Electronic Devices and Circuit Theory*. 7th ed. New Jersey: Prentice Hall.

Gadre, D.V. 2001. *Programming and Customizing the AVR Microcontroller*. 1st ed. New York: McGraw-Hill. doi:10.1036/00713997.

Goldie, John. 1998. "Summary of Well Known Interface Standard." http://ecee.colorado.edu/~mcclurel/nan216.pdf.

Hsiung, S.C. 2007. "The Use of PIC Microcontrollers in Multiple DC Motors Control Applications." *Journal of Industrial Technology* 23 (3): 1–9.

Khotimah, Purnomo H., and Dikdik Krisnandi. 2010. "Komunikasi Antar Mikrokontroler : *Master-Slave* Menggunakan ATMega 8535 Pada Pengembangan Sistem Stasiun Cuaca." *Prosiding Seminar Nasional Teknoin 2010: Pengembangan Teknologi Industri Berbasis "Green Technology,"* C81–86.

Krisnandi, Dikdik. 2011. "Perancangan Dan Analisa Output Rangkaian Signal Conditioning Analog Melalui Mikrokontroler ATMega8535 Untuk Stasiun Cuaca." *Inkom: Jurnal Informatika, Sistem Kendali Dan Komputer* V.

Sankar, K Prajindra, S K Tiong, and S P Johnny Koh. 2009. "Microcontroller System for Adaptive Antenna Downlink Transmitter Power Optimization." *International Journal of Electrical and Computer Engineering* 3 (2): 604–8.